DaeMon: Architectural Support for Efficient Data Movement in Fully Disaggregated Memory Systems

Christina Giannoula
Kailong Huang, Jonathan Tang, Nectarios Koziris, Georgios Goumas, Zeshan Chishti, Nandita Vijaykumar
Executive Summary

Problem:
Efficient data movement support is a major system challenge for fully Disaggregated Systems (DSs)

Contribution:
DaeMon: the first adaptive data movement solution for fully DSs

Key Results:
DaeMon achieves 2.39x better performance and 3.06x lower data access costs over the widely-adopted scheme of moving data at page granularity
What is resource disaggregation?
Monolithic vs Disaggregated Systems

thanks to recent advances in network technologies
Benefits of Fully Disaggregated Systems

• Resource Utilization
• Failure Handling
• Resource Scaling
• Heterogeneity

Disaggregated systems can significantly decrease data center costs
Baseline Disaggregated System

Network

Compute Component

Memory Component

Compute Component

Memory Component

Compute Component

Memory Component

CPU

Local Memory

Controller

Remote Memory
Baseline Disaggregated System

- **Network**
  - **Compute Component**
  - **Memory Component**
  - **Compute Component**
  - **Memory Component**
  - **Compute Component**
  - **Memory Component**
  - **Remote Memory**
  - **Local Memory**

- **CPU**

- **Controller**

- Hosts ~20% of application’s data
Baseline Disaggregated System

- Compute Component
- Memory Component

Network

CPU
- Local Memory

Controller
- Remote Memory

hosts ~80% of application’s data
Baseline Disaggregated System

- **CPU**
- **Local Memory**
- **Remote Memory**
- **Controller**

Data is typically moved at page granularity.
Baseline Disaggregated System

Network

- Compute Component
- Compute Component
- Compute Component
- Distributed OS modules

- CPU
  - Local Memory

- Controller
  - Remote Memory

- Memory Component
  - Memory Component
  - Memory Component
  - Memory Component
Why is data movement challenging?
#1: Coarse-Grained Data Migrations

- Page granularity (e.g., 4KB) data migrations:
  - Software transparency
  - Low metadata overheads
  - High spatial locality

Diagram showing network, CPU, local memory, controller, and remote memory.
#2: Non-Conventional System Design

- Disaggregated systems are **not monolithic**

- Hybrid/heterogeneous memory systems:
  - Thermostat [ASPLOS’17]
  - Kleio [HPDC’19]
  - Chameleon [MICRO’18]
  - HSCC [ICS’17]
  - Nimble [ASPLOS’19] …

- Distributed memory management

- Centralized memory management
#2: Non-Conventional System Design

- Disaggregated systems are **not monolithic**
  - Hybrid/heterogeneous memory systems:

  - Hardware units in the CPU side would incur high hardware overheads

- **System-Level Solutions**
  - Controller
  - Remote Memory

- **Hardware-Level Solutions**
  - Chop [HPCA’10]
  - UH-MEM [CLUSTER’17]
  - MemPod [HPCA’17]
  - LGM [IPDPS’19] ...
#2: Non-Conventional System Design

- Disaggregated systems are **not monolithic**

  ![Diagram showing CPU, Controller, Local Memory, Remote Memory]

  - Hybrid/heterogeneous memory systems:

    Prior solutions are not **suitable** or **efficient** for disaggregated memory systems
#3: Variability in Data Access Latencies

- Data access latencies depend:
  - **Location** of the remote memory component

![Diagram showing the location of remote memory components](image)

- different locations for application’s data
#3: Variability in Data Access Latencies

- Data access latencies depend:
  - Location of the remote memory component
  - Network contention
How can we build an efficient solution?
1. Disaggregated Hardware Support

- **Independence**
- **High Parallelism**
- **High Scalability**

![Diagram of Compute Component and Memory Component with dedicated units highlighted.](image)
2. Multiple Granularity Data Movement

- **Compute Component**
  - DaeMon Compute Engine
  - CPU
  - LLC
  - Local Memory
  - Sub-block Queue
  - Page Queue

- **Memory Component**
  - DaeMon Memory Engine
  - Controller
  - Remote Memory
  - Sub-block Queue
  - Page Queue
2. Multiple Granularity Data Movement

Prioritization of cache line migrations
2. Multiple Granularity Data Movement

- Software Transparency
- Low Metadata Overheads
- High Spatial Locality
- Latency-Efficiency in Critical Data
3. Link Compression in Page Migrations

Compute Component
- DaeMon Compute Engine
- CPU
- LLC
- Local Memory
- Sub-block Queue
- Page Queue
- (De) Compr. Unit

Memory Component
- DaeMon Memory Engine
- Controller
- Remote Memory
- Sub-block Queue
- Page Queue
- (De) Compr. Unit

Cache lines
Compressed pages

compressed pages inside the network
3. Link Compression in Page Migrations

- **Compute Component**
  - DaeMon Compute Engine
  - CPU
  - LLC
  - Sub-block Queue
  - Page
  - Queue Controller
  - Compression Unit
  - Page Queue Controller (De) Compr.
  - DaeMon Compute Engine

- **Memory Component**
  - DaeMon Memory Engine
  - Controller
  - Sub-block Queue
  - Page
  - Queue Controller
  - Cache lines Compressed

- ✓ Bandwidth-Efficiency
- ✓ Critical Cache Line Prioritization
4. Selection Granularity Data Movement

Compute Component
DaeMon Compute Engine

- CPU
- LLC
- Local Memory
- Sub-block Queue
- Page Queue
- Queue Controller
- Cache line, page or both?
- (De) Compr. Unit

Memory Component
DaeMon Memory Engine

- Controller
- Remote Memory
- Sub-block Queue
- Page Queue
- Queue Controller
- (De) Compr. Unit
4. Selection Granularity Data Movement

DaeMon Compute Engine

- CPU
- LLC
- Local Memory
- Sub-block Queue
- Page Queue
- Inflight Sub-block and Page Buffers
- (De) Compr. Unit
- Queue Controller

DaeMon Memory Engine

- Controller
- Remote Memory
- Sub-block Queue
- Page Queue
- (De) Compr. Unit
- Queue Controller

Cache lines, pages or both?
4. Selection Granularity Data Movement

- Robustness
- Versatility
- Adaptivity to Runtime Changes
Why does this work?
Use Case 1: Memory Access Patterns

Compute Component
- CPU
- LLC
- DaeMon Engine
- Inflight Buffers
- Selection Gran. Unit
- Local Memory

Memory Component
- Controller
- DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization
- Sub-block Page
- Sub-block Page
- Sub-block Page
- Sub-block Page
- Sub-block Page

high locality within pages

Time

30
Use Case 1: Memory Access Patterns

Compute Component

CPU
LLC

DaeMon Engine
Selection Gran. Unit

Inflight Buffers

Inflight Buffers Utilization

Local Memory

Memory Component

Controller
DaeMon Memory Engine
Remote Memory

Cache lines
Compressed pages

Selection Gran. Unit

Inflight Buffers

Sub-block
Page

Sub-block
Page

Sub-block
Page

Sub-block
Page

...
Use Case 2: Network Characteristics

- **CPU**
  - LLC
  - Inflight Buffers
  - Selection Gran. Unit

- **Local Memory**

- **DaeMon Engine**
  - Compute Component

- **Remote Memory**

- **Inflight Buffers**
  - Utilization
  - Sub-block
  - Page

- **Controller**
  - DaeMon Memory Engine

- **Memory Component**

**Horizontal Axis:**
- 32 Time

**Vertical Axis:**
- high bandwidth consumption
Use Case 2: Network Characteristics

Compute Component
- CPU
- LLC
- DaeMon Engine
  - Inflight Buffers
  - Selection Gran. Unit
- Local Memory

Memory Component
- Controller
  - DaeMon Memory Engine
- Remote Memory

Inflight Buffers Utilization
- Sub-block
- Page

High bandwidth consumption
Low bandwidth consumption

Gran. Unit
Speedup in Real Applications

- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon
- DaeMon-Compr

Workloads: kc, tr, pr, nw, bf, bc, ts, sp, sl, hp, pf, dr, rs, GM

Speedup: 0, 1, 2, 3, 4, 5

Graph showing speedup for various workloads.
Speedup in Real Applications

- Page
- ComprPage
- CacheLine

CacheLine+Page
DaeMon-Compr
DaeMon

Workloads:
- kc
- tr
- pr
- nw
- bf
- bc
- ts
- sp
- sl
- hp
- pf
- dr
- rs
- GM

Speedup:
- 14.6
- 1.95x
Speedup in Real Applications

- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon-Compr
- DaeMon

Speedup:
- 14.6
- 1.29x

Workloads:
- kc
- tr
- pr
- nw
- bf
- bc
- ts
- sp
- sl
- hp
- pf
- dr
- rs
- GM
Speedup in Real Applications

- Page
- ComprPage
- CacheLine
- CacheLine+Page
- DaeMon-Compr
- DaeMon

Speedup:
- kc: 8.4x
- tr: 14.6x
- pr: 0.95x
- nw: 1.09x
- bf: 1.09x
- bc: 1.09x
- ts: 1.09x
- sp: 1.09x
- sl: 1.09x
- hp: 1.09x
- pf: 1.09x
- dr: 1.09x
- rs: 1.09x
- GM: 1.09x

Workloads: kc, tr, pr, nw, bf, bc, ts, sp, sl, hp, pf, dr, rs, GM
DaeMon performs best in real-world applications.
Conclusion

• Data movement is a major challenge for fully DSs
• Prior solutions are not suitable or efficient
• DaeMon is the first adaptive data movement solution
• DaeMon consists of four techniques:
  • Disaggregated hardware support
  • Decoupled multiple granularity data movement
  • Link compression in page movements
  • Selection granularity data movement
• DaeMon’s benefits over the widely-adopted scheme:
  • 2.39x better performance
  • 3.06x lower data access
• DaeMon is highly-efficient, low-cost, scalable and robust
DaeMon: Architectural Support for Efficient Data Movement in Fully Disaggregated Memory Systems

Christina Giannoula
Kailong Huang, Jonathan Tang, Nectarios Koziris, Georgios Goumas, Zeshan Chishti, Nandita Vijaykumar

Thank you!